Cold Start

SEMINAR RECOMMENDER SYSTEMS WS16/17

NIKLAS (234893), SUPREETHA (271416), LUCKY (276785)

Structure

1. Introduction
2. Recommending new Movies: Even a Few Ratings Are More Valuable Than Metdata
3. Pairwise Preference Regression for Cold-start Recommendation
4. Cross-Domain Collaborative Filtering Over Time
5. Differences and similarities
6. Winning method

Recommender Systems

- Use opinions and preferences of users to help others find useful and interesting content
- Personalize what users see

Philosophy:

- Makes use of the social process of friends endorsing items as recommendations to others in their community. Recommender systems automate this process.

Applications:

- Offering news articles to on-line newspaper readers, based on a prediction of reader interests.
- Online e-commerce applications like Amazon to improve user engagement and loyalty

Content based filtering (CBF)

Working:

- In this approach, recommendations are made by making use of the historical data comprising of previous ratings given by the user. This rating data is used to draw a profile of the user.

Advantage:

- Major benefit is that it can be used to make recommendations to new items in the absence of historical data. This absence is compensated by making the users answer a questionnaire from which their respective profiles can be drawn and then further used.
- Recommend users new items which have not been rated

Disadvantage:

- The drawback of this approach is the inherent lack of diversity in the recommendations made. Recommended items are similar to the ones made previously.
- The performance of this approach highly depends on the quality of feature generation and selection

Example for CBF

Name	Genre	Director	Actor
Insidious	Horror	James Wan	Parick Wilson
Shutter Island	Thriller	Martin Scorsese	Leonardo DiCaprio

Film Metadata

Name	Genre	Director	Actor
	Thriller, Drama	Martin Scorsese	Patrick Wilson, Bruce Willis

Preference profile of an user

Collaborative filtering (CF)

Working :

- Associates a user with a group of like-minded users
- Recommend items enjoyed by other users in the group

Advantage:

- Content-Independent
- Captures hidden connection
- Performs better than CBF when a lot of ratings on items or users are given
- Provides ‘Serendipitous findings’ which is absent in CBF.

Disadvantage:

- Suffers from ‘Cold Start’ problems. When historical data on items or users in not available, the performance becomes questionable.

Key challenge for cold start scenarios

Providing high quality recommendation to users in ‘cold start’ situations.

3 types of cold start scenarios:

- Recommendation of new items to existing users
- Recommendation of existing items to new users
- Recommendation of new items to new users.

Aim of the presentation:

- Present solutions to overcome these different cold start scenarios

Cold start problems

	Existing Users	New Users
Existing Items	Partition I	Partition II
New Items	Partition III	Partition IV

Data that could be used to solve cold start problems:

- Item-and user-metadata
- Historical ratings from another domain

Cold start vs. sparsity

- **Data sparsity** arises from the phenomenon that users in general rate only a limited number of items
- **Cold start** refers to the difficulty in bootstrapping the Recommender System for new users or new items.

Matrix Factorization (MF) for cold start

- For a scenario where there are new users and new items, matrix factorization can only generate the predicted ratings but lacks the historical ratings to know the loss/ difference in your predictions

Recommending New Movies: Even a Few Rating are More Valuable Than Metadata

NIKLAS MELCHER (234893)

Structure

1. Motivation
2. Hypotheses
3. First part: Movie-NSVD1
    1. State of the art
    2. Movie-NSVD1
    3. Experiment – Movie-NSVD1 vs. Matrix Factorization
4. Second part: Predicting ratings on new movies
    1. Experiment – Predicting ratings on new movies
5. Conclusion

Motivation

First part: Movie-NSVD1

- Propose an generalized algorithm to build a bridge between CF and CBF
- Compare methods for meta-data based methods to rating-based ones

Second part: Predicting ratings on new movies

- Investigate for what number of ratings meta-data based methods are better than rating ones

Hypotheses

First part: Movie-NSVD1

- Prediction performance of CBF movie-NSVD1 is comparable to that of MF for CF

Second part: Predicting ratings on new movies

- „Even a few ratings are more valuable than metadata“
- But Metadata based methods are better when there are very less ratings for new movies

First Part: Movie-NSVD1

State of the art – MF

- The goal of MF: approximate $R \in \mathbb{R}^{N\ x\ M}$ as a product of $P \in \mathbb{R}^{N\ x\ K}$ and Q $\in \mathbb{R}^{M\ x\ K}$ $(r_{ui} = p_u^T * q_i)$
- Minimize error of prediction, $e_{ui} = r_{ui} - \hat{r}_{ui}$ while keeping the Euclidian norm of the user and movie feature vectors small
  - $(P^*, Q^*) = arg_{P,Q} \min \sum_{(u,i)\in R}(e_{ui}^2 + \lambda p_u^T p_u + \lambda q_i^T q_i)$

State of the art – Learning algorithm

Learning algorithm:

- Alternating Least Squares
- (Stochastic) Gradient Descent

BRISMF approach (based on stochastic gradient descent):

1. Order data set by user id
2. Order then by rating date
3. Update model per-rating (not per-user or movie)
4. $p'_u = p_u + \eta * (e_{ui} * q_i - \lambda * p_u)$
5. $q'_i = q_i + \eta * (e_{ui} * p_u - \lambda * q_i)$

State of the art – NSVD1

NSVD1:

- Prediction is made by the following rule: $\hat{r}_{ui} = b_u + c_i + p_u^T * q_i$
- Where $p_u = (\frac{1}{\sqrt{n_u}} * \sum_{j:(u,i)\in R} w_j)$
- n_u number of ratings of user
- p_u is a function of other variables (MF, it is arbitrary)
- b_u and c_i are user and movie biases
- w_j vectors are secondary movie feature vectors (there are two sets of movie vectors)
- → Fully metadata based algorithm

Movie-NSVD1 – Training algorithm

- Users are represented by M dimensional binary vectors
  - 0 not rated, 1 rated
  - Infer p_u by a linear transformation, denoted by W
- Learning algorithm finds W and Q
- Run NSVD1 interchanging role of users and movies
- Movie-NSVD1

Movie-NSVD1 – Notation

- $W \in \mathbb{R}^{C\ x\ K}$ is the transformation matrix that transforms movie metadata to feature vectors
- $w_l \in \mathbb{R}^{K\ x\ 1}$ is transpose of the l-th row of W
- $X \in \mathbb{R}^{N\ x\ C}$ contains metadata about the movie → Fully specified
- $x_i \in \mathbb{R}^{C\ x\ 1}$ is the transpose of the i-th row of X
- x_{il} is the (i, l) −th element of X

Movie-NSVD:1 Training algorithm

```
1  Create random model.
2  repeat
3      one epoch:
4      for i ← 1 to M do
5          q_i ← W^T x_i
6          a ← q_i
7          foreach u : (u, i) ∈ R do
8              r̂_ui ← b_u + c_i + p_u^T q_i
9              e_ui ← r_ui − r̂_ui
10             b ← p_u
11             p_u ← p_u + η · (e_ui · q_i − λ · p_u)
12             q_i ← q_i + η · (e_ui · b − λ · q_i)
13             b_u ← b_u + η · (e_ui − λ · b_u)
14             c_i ← c_i + η · (e_ui − λ · c_i)
15         end
16         d ← 1/(x_i^T x_i)
17         foreach l : x_il ≠ 0 do
18             w_l ← w_l + d · x_il · (q_i − a)
19         end
20     end
21 until RMSE on the test set decreases ;
```

Algorithm 1: Training algorithm for movie-NSVD1

for every movie

Compute movie feature vector

for every user in given set of ratings

Update with incremental gradient descent

Hyperparameter: Allow x_{il} to be arbitrary

For every row of W

Backpropagate the change in q_i to W

Time requirement: $O(K * |X| + K * |R|)$

Generalized approach for NSVD1

- **Idea:**
  - Allow using both user and movie metatdata
  - Not all user and/or item features depend on metadata
- $k \in \{K1, \ldots, K2\}$: metadata for user features
- $k \in \{K3, \ldots, K4\}$: metadata for item features
- $P[1,..N; K1..K2]$: Submatrix of P
- $Q[1,..M; K3..K4]$: Submatrix of Q
- X‘ and X“: Matrix of user-and item- metadata
- W‘ and W“: used to recompute submatrices of P and Q from metadata

Generalized NSVD1

```
1  Create random model.
2  repeat
3      one epoch:
4      P[1..N; K1..K2] ← X'W'
5      Q[1..M; K3..K4] ← X''W''
6      foreach (u, i) ∈ R do
7          update b_u, c_i, p_u and q_i as in
8          Algorithm 1, lines 8–14.
9      end
10     for 1 to n2 do
11         for u ← 1 to N do
12             e ← P[u; K1..K2]^T − W'^T x'_u
13             d ← 1/(x'_u^T x'_u)
14             W' ← W' + η2 · (d · x'_u · e^T − λ2 · W')
15         end
16     end
17     P[1..N; K1..K2] ← X'W'
18     for 1 to n2 do
19         for i ← 1 to M do
20             e ← Q[i; K3..K4]^T − W''^T x''_i
21             d ← 1/(x''_i^T x''_i)
22             W'' ← W'' + η2 · (d · x''_i · e^T − λ2 · W'')
23         end
24     end
25     Q[1..M; K3..K4] ← X''W''
26 until RMSE on the test set decreases ;
```

Algorithm 3: Batch training algorithm for a generalized NSVD1

compute submatrices

for every user

update transformation matrix for user features

update user feature submatrix

for every item

update transformation matrix for item features

update item feature submatrix

New parameters:

- η_2: learning rate
- λ_2: regularization factor
- n_2: iteration count

Experiment – MF vs. Movie-NSVD1

Datasets	**Probe**	**Probe10**	**Rest of Probe 10**	**Quiz**
Kind of dataset	Testset	Testset	Trainset	testset
#ratings	1 408 395	140 840	1 267 556	2 817 131/2 (ratings are withheld)

- Collect movie metatdata and represent texts as vectors
- Use following zones: title, synopsis, actor, director, release year, genre.
- Movies are described with 146 810 dimensional vectors
- Dimensional vectors e.g.: genre:comedy, title:comedy
- 81 nonzero values on average

Learning algorithms for MF

Learning Method	Description	RMSE
IGD01	Incremental gradient descent with bias b_u and c_i	0,9079
AG01	Like **IGD01**, but with alternating gradient descent	0,9096
ALS01	MF with biases b_u and c_i, using alternating least squares, starting with movie features	0,9257

Description of subsets for Movie-NSVD1

Name	Description	Dimensional vectors	Nonzero features on average
T1	keeping all the features	146 810	81
T2	keeping only those features that occur in least 2 movies	64 746	76
T4	At least 4 movies	33 838	72
T10	At least 10 movies	14 547	66
T20	At least 20 movies	7340	60

Learning algorithms for Movie-NSVD1

Name	Description
IGD-I: IGD	With incremental backpropagation: Using movie-ordered database
AGD-I: AGD	With incremental backpropagation
IGD-B: IGD	With batch-backpropagation. Using (user, time)-ordered database
AGD-B: AGD	With batch-propagation

For B variants:

- $n_2 = 1$
- $n_2 = 10$

Results

	n_2	T1	T2	T4	T10	T20
IGD-I	–	0.9143	0.9155	0.9161	0.9200	0.9285
IGD-B	1	0.9121	0.9130	0.9147	0.9219	0.9370
IGD-B	10	0.9089	0.9093	0.9101	0.9157	0.9322
AGD-I	–	0.9165	0.9175	0.9186	0.9224	0.9306
AGD-B	1	0.9148	0.9163	0.9184	0.9265	0.9401
AGD-B	10	0.9112	0.9117	0.9120	0.9170	0.9312
ALS-B	1	0.9429	0.9461	0.9509	0.9616	0.9706
ALS-B	10	0.9327	0.9350	0.9389	0.9608	0.9716

Table 1: Probe10 RMSE of movie-NSVD1 with different metadata and learning methods

Recall pure MF:

- IGD01 - RMSE: 0,9079
- AG01 - RMSE: 0,9096
- ALS01 - RMSE: 0,9257

Interpretation of the results

- More metadata imply better prediction performance
- More metadata means less interdependency between movies because movie discriptions are dissimiliar
- But pure matrix factorization, using no metadata, still aims better performance

Part 2: Predicting ratings on new movies

Predicting ratings on new movies

- CBF has advantage over CF: ability to handle new movies
- New movies have no ratings
- But metadata is available

Setup

- Createt 10 partitions for movies, then applied 10-fold Cross-Validation
  - Split training set into train subset and test subset
  - Compare different prediction functions on subsets
- For each partition: Train a model with 9/10 of the original data (all ratings except Probe10)
- Evaluated the model on 1/10 of the original test data (Probe10)
- RMSE: Sum of squared errors of each of the 10 evaluations → X10 RMSE

Experiment

- Compare the movie-NSVD1 with user-bias-based methods
- User-bias-based methods:
  1. User-average predictor: $\hat{r}_{ui} = b_u$
  2. Bias predictor: $\hat{r}_{ui} = b_u + c_i, b_u$ & $c_i$will be re-estimated. $c_i = 0$ for new items
  3. User-bias predictor: : $\hat{r}_{ui} = b_u$ again, but use incremental gradient method to give larger weights to newer examples
  4. Input-bias-predictor: $\hat{r}_{ui} = b_u + w^T * x_i$, where b_u and w are estimated using ALS.

Results

Predictor	RMSE
Movie-NSVD1	1,0080
User-average predictor	1,0616
Bias predictor	1,6015
User-bias predictor	1,0504
Input-bias predictor	1,305
Simple bias predictor of Probe10 (many ratings)	0,9700 – 0,9900

→ When we know the movie average, c_i(the movie bias), it is much more valuable than any metadata

When are ratings more valuable?

- Compare two methods:
  1. Movie-NSVD1 with cross-validation procedure
  2. Bias predictor (user- and movie-bias) with incremental gradient method, where first N ratings of movies were included in training set
- When will RMSE of bias predictor will pass over RMSE of movie-NSVD1?

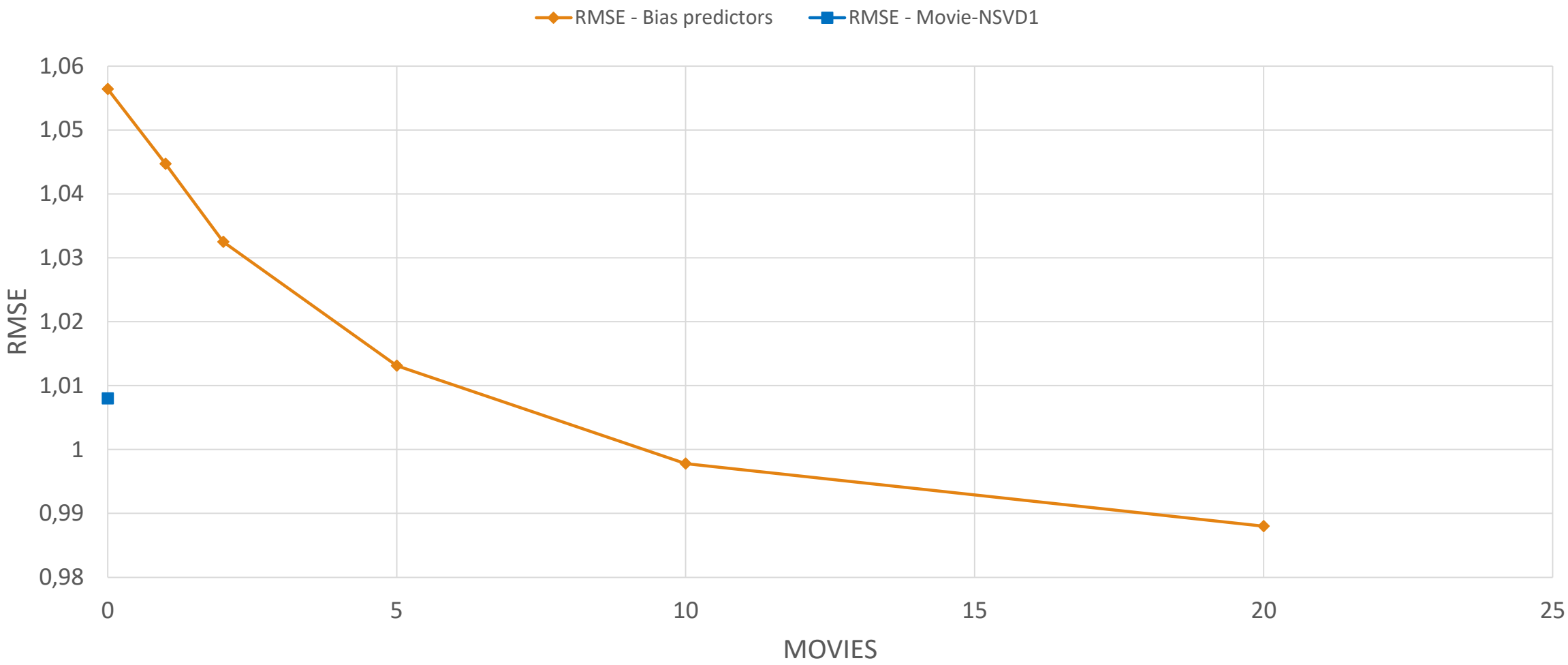
Result

- Recall: RMSE of movie-NSVD1 → 1,0080

N	0	1	2	5	10	20
RMSE	1.0564	1.0447	1.0325	1.0131	0.9978	0.9880
N	30	40	50	100	200	500
RMSE	0.9841	0.9819	0.9802	0.9772	0.9754	0.9740

Table 2: X10 RMSE of bias predictors, when N ratings from each skipped movie are added to the training set

MOVIE-NSVD1 VS. SIMPLE BIAS PREDICTOR



Final conclusion

- Authors presented movie metatdata based approach: movie-NSVD1
- Generalzed this algorithm to handle user and movie-metadata
- Showed that more metadata imply better prediction performance
- **But:** Showed that even 10 ratings are more valuable than any metadata

→People rate movies, not descriptions

References

Pilászy, István, and Domonkos Tikk. "Recommending new movies: even a few ratings are more valuable than metadata." *Proceedings of the third ACM conference on Recommender systems*. ACM, 2009.

Jannach, Dietmar ; Zanker, Markus ; Felfernig, Alexander ; Friedrich, Gerhard: Recommender Systems : An Introduction. Cambridge: Cambridge University Press, 2010.

Takács, Gábor, et al. "A unified approach of factor models and neighbor based methods for large recommender systems." *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the*. IEEE, 2008.

Paterek, Arkadiusz. "Improving regularized singular value decomposition for collaborative filtering." *Proceedings of KDD cup and workshop*. Vol. 2007. 2007.

Pairwise Preference Regression for Cold-start Recommendation

SUPREETHA SUDHEENDRA

Structure

1. Motivation
2. Hypothesis
3. State of the art
4. Proposed Method
5. Methodology
    1. Profile Construction
    2. Design
6. Pseudocode
7. Experiments
8. Advantages
9. Disadvantages
10. References

Motivation

- Cold-Start is a scenario that exists perpetually in RSs and require to be addressed with a robust design and solution
- Several algorithms have been designed which make use of either of the two approaches to the designing of Recommender Systems.
- Several algorithms have been designed which make use of approaches that is a hybrid of CBF (Content Based Filtering) and CF(Collaborative Filtering) *but* with the need for better quality recommendations especially during cold start which is a key challenge in Recommender Systems.
- Focus is specially placed on providing reasonable recommendations of new items to new users.

Hypothesis

In a nutshell:

- Content Based Filtering: Places importance on the ratings given by users
- Collaborative Filtering: Places importance on the features of users and items. Ratings are treated as user profiles in-order to evaluate the commonalities between them.
- To attack the problem of cold-start the authors propose a new *hybrid approach* exploiting both user and item features.
- The idea here is to collect three different types of information and leverage them: User features, Item Features and Ratings given by users.
- These Ratings are utilized as *targets* that reveal the affinity between the users features and item features.

State of the art

Some of the concepts that the paper makes use of is :

- Balabanović, Marko, and Yoav Shoham. "Fab: content-based, collaborative recommendation." *Communications of the ACM* 40.3 (1997): 66-72. Fab is the first hybrid algorithm which builds user profiles based on content analysis and calculates user similarities based on these profiles.
- Good, Nathaniel, et al. "Combining collaborative filtering with personal agents for better recommendations." *AAAI/IAAI*. 1999.
- Chu and Park proposed a predictive bilinear regression model in 'Dynamic content environment', where the popularity of items changes temporally, lifetime of items in very short and recommender systems are forced to recommend only new items.
- This work suggests to maintain profiles for both users and items where temporal characteristics of contents like popularity are updated in real-time. Therefore at a time the number of available items are less

Proposed Method

- Construct tensor profiles for user/item pairs from their individual features. Within the tensor regression framework, we optimize the regression coefficients by minimizing *pairwise preference loss.*
- **Goal**: To make reasonable recommendations to new users with no historical ratings but just a few demographic information.

What is Ranking?

- Learning to rank or machine-learned ranking (MLR) is the application of machine learning, typically supervised, semi-supervised or reinforcement learning, in the construction of ranking models for information retrieval systems.
- Training data consists of lists of items with some partial order specified between items in each list. This order is typically induced by giving a numerical or ordinal score or a binary judgment (e.g. "relevant" or "not relevant") for each item.

Approaches:

- Pairwise : In this case learning-to-rank problem is approximated by a classification problem — learning a binary classifier that can tell which document is better in a given pair of documents. The goal is to minimize average number of inversions in ranking.

Methodology

- Split user ratings into 4 partitions, users and items into new and existing.
- Partition I – Training Dataset; Partition II,III, IV – Test Dataset
- User declared demographic information available includes: Age, Gender, Residence
- Item Information when items are either created or acquired, include: Product Name, Manufacturer, Genre, Production year
- **Key Idea**: Build a predictive model for user/item pairs by leveraging all the available information on users and items to attack cold start situations especially with recommending *new items to new users*.
- We divide the methodology into **Profile construction** and Algorithm **Design**

Profile construction

- In this paper, each item is represented by a set of features, denoted as a vector z, where $z \in R^D$ and D is the number of item features, Similarly, each user is represented by a set of user features, denoted as x, where $x \in R^C$ and C is the number of user features.
- Note that we append a constant with no information is represented as [0,…,0,1] instead of a vector of zero entries for each user.

Profile construction

- In traditional collaborative filtering (CF), the ratings given by users on items of interest are used as user profiles to evaluate commonalities between users.
- In our regression approach, we *separate these feedbacks* from user profiles. The ratings are utilized as *targets* that reveal affinities between user features to item features.

Profile construction

In the paper, we have collected three sets of data, including item features, user profiles and the ratings on items given by users. Let the u-th index of user be represented as x_u ,

- The i-th content item as z_i and
- denote by r_{ui} the interaction between the user x_u and the item z_i
- Let Θ denote the index set of observations $\{r_{ui}\}$

Design

- A predictive model relates a pair of vectors X_u and Z_i to the rating r_{ui} on the item given by the user. There are various ways to construct a joint feature space for user/item pairs.
- We focus on the representation as outer products $\mathbf{x}_u \otimes \mathbf{z}_i$, a vector of CD entries $\{x_{u,a} z_{i,b}\}$ where $z_{i,b}$ denotes the b-th feature of Z_i and $X_{u,a}$ denotes the a-th feature of X_u

Design

Regression on Pairwise Preference

We define a parametric indicator as a bilinear function of X_u and Z_i in the following:

$$s_{ui} = \sum_{a=1}^{C} \sum_{b=1}^{D} x_{u,a} z_{i,b} w_{ab}$$

where C and D are the dimensionality of user and content features respectively, a, bare feature indices. The weight variable is independent of user and content features and characterizes the affinity of these two factors in interaction.

Design

The regression coefficients can be optimized in regularization framework, i.e.

$$\arg\min_{w} \sum_{ui \in \Theta} (r_{ui} - s_{ui})^2 + \lambda \| w \|^2$$

where λ is a tradeoff between empirical error and model complexity.

Design

In this paper, we introduce a personalized pairwise loss in the regression framework. For each user x_u ,the loss function is generalized as

$$\frac{1}{n_u}\sum_{i\in\Theta_u}\sum_{j\in\Theta_u}((r_{ui}-r_{uj})-(s_{ui}-s_{uj}))^2$$

where Θ_u denotes the index set of all items the user x_u has rated,

the number of ratings given by the user x_u

$n_u = |\Theta_u|$

Design

Replacing the squares loss by the personalized pairwise loss in the regularization framework, we have the following optimization problem:

$$\min_{w} \sum_{u} \left(\frac{1}{n_u} \sum_{i \in \Theta_u} \sum_{j \in \Theta_u} \left((r_{ui} - r_{uj}) - (s_{ui} - s_{uj}) \right)^2 \right) + \lambda \| w \|^2$$

where u runs over all users. The optimal solution can be computed in a closed form as well, i.e.

$$w^* = \left(A + \frac{\lambda}{2} I \right)^{-1} B$$

Design

$$A = \sum_{u} \sum_{i \in \Theta_u} z_i (z_i - \overline{z_u})^T \otimes x_u x_u^T$$

$$B = \sum_{u} \sum_{i \in \Theta_u} r_{ui} (z_i - \overline{z}_u) \otimes x_u$$

$$\overline{z}_u = \frac{1}{n_u} \sum_{i \in \Theta_u} z_i$$

Pseudo-Code

1. Procedure: Procedure: Pairwise Preferance Regression for Cold-start Recommendation

Input: x_u x ∈ R^C , z_i $z \in R^D$

2. Initialize j, j ∈ Z
3. For $x_u$in X do
3. For z_i in Z do
4. for x_a in C
5. for z_b in D
6. Initialize w
7. repeat
8. w: = w- λ(w*)
9. until convergence
10. return arg min (w)

Every user ‘u’ represented as a vector

Every item ‘i’ represented as vector in Z

Loop through every attribute of user ‘u’

Loop through every attribute of item ‘i’

Initialize the value of weight variable that we need to optimize

$$\mathbf{w}^* = \left(\sum_{ui \in \mathbb{O}} \mathbf{z}_i \mathbf{z}_i^\top \otimes \mathbf{x}_u \mathbf{x}_u^\top + \lambda \mathbf{I} \right)^{-1} \left(\sum_{ui \in \mathbb{O}} r_{ui} \, \mathbf{z}_i \otimes \mathbf{x}_u \right)$$

11. Compute S

$$\min_{\mathbf{w}} \sum_u \left(\frac{1}{n_u} \sum_{i \in \mathbb{O}_u} \sum_{j \in \mathbb{O}_u} ((r_{ui} - r_{uj}) - (s_{ui} - s_{uj}))^2 \right) + \lambda \|\mathbf{w}\|_2^2$$

12. return S
13. End procedure

Pseudocode- using closed form

1. Procedure: Pairwise Preferance Regression for Cold-start Recommendation

   Input: x_u x ∈ R^C , z_i $z \in R^D$

2. For u=1,2,.....X do
3. For i=1,2....D do
4. $\check{z} = \frac{1}{n_u}\sum_{i \in O_u} z_i$

Loop through every user in X

Loop through every item in Z

Calculate the average rating given by this user

5. $$\mathbf{A} = \sum_{u} \sum_{i \in \mathbb{O}_u} \mathbf{z}_i (\mathbf{z}_i - \tilde{\mathbf{z}}_u)^\top \otimes \mathbf{x}_u \mathbf{x}_u^\top$$
6.
7. $$\mathbf{B} = \sum_{u} \sum_{i \in \mathbb{O}_u} r_{ui} (\mathbf{z}_i - \tilde{\mathbf{z}}_u) \otimes \mathbf{x}_u$$

$$\mathbf{w}^* = \left(\mathbf{A} + \frac{\lambda}{2} \mathbf{I} \right)^{-1} \mathbf{B}$$

8. return |w|= $||w||_2$

Return the norm of w which can be

9. End procedure

Experiments

Competitive Approaches

- Most popular
- Segmented most popular
- Vibes Affinity 1
- Vibes Affinity 2

Note: In this approach we evaluate the performance of the recommender system based on the correctness of ranking rather than prediction accuracy.

Experiments

Most popular

Most Popular (MP) provides the same recommendation to all users based on *global popularity* of items. The global popularity of an item is measured as following :

$$s_i = \frac{\overline{r_i} * n_i + \overline{r} * \alpha}{n_i + \alpha}$$

Where, Avg $\overline{r_i}$ is defined as $\frac{1}{n_i}\sum_{u \in O_i} r_{ui}$ the support $n_i = |O_i|$ is the number of users who have rated the i-th item,

$\overline{r}$ denotes the average of all ratings and α is the shrinkage parameter

$\overline{r}$ for MovieLens= 3.6 and for EachMovie= 4.32

Experiments

Segmented most popular

Segmented Most Popular (SMP) divides users into several user segments based on user features (i.e. age or gender) and computes predictions of items based on their local popularity within the user segment which a target user belongs to :

$$s_{ci} = \frac{\overline{r_{ci}} * n_{ci} + \bar{r} * \alpha}{n_{ci} + \alpha}$$

Where, $\overline{r_{ci}}$ is the average rating of an item i within a user segment c , n_{ci} is the number of users who have rated i within the segment c and α is the shrinkage

Experiments

Vibes Affinity 1 (Partition II)	Vibes Affinity 2 (For Partition III, IV)
$s_{uj} = \sum_{i \in I_u} f(i \rightarrow j)$	$s_{uj} = \sum_{a \in A_u} \sum_{f \in F_j} f(a \rightarrow j)$
Where, I_u is the set of items consumed by user 'u' i , j are items f(i->j) is the affinity between item i and item j S_{uj} is the preference score of each item j for a user 'u'	Where, a is a user-attribute f is an item feature A_u is the set of attributes that user 'u' has F_j is the set of features the item 'j' has

Experiments : Performance Metric

Cumulative Gain : It is the sum of graded relevance values of all results in a search result list.

$$CG = \sum_{i=1}^{P} rel(i)$$

Where, rel(i) : Graded relevance of the result at position i.

Relevance : how well a retrieved document or set of documents meet the information need of the query.

Discounted Cumulative Gain is a measure of ranking quality

$$DCG = rel(i) + \sum_{i=2}^{p} rel(i)/log2(i)$$

Normalized DCG: The search result lists vary in length depending on the query . Therefore the gain at every position for a chosen value of p should be normalized across queries.

nDCG takes a value between o and 1. nDCG at position p is given by,

$$nDCG = DCG/IDCG$$

Experiments

Measurement

We evaluate performance of recommender system based on the correctness of ranking rather than prediction accuracy, the normalized Discounted Cumulative Gain (nDCG), widely used in information retrieval (IR), as performance metric.

$$nDCG_k = \frac{1}{|U_T|} \sum_{u \in U_T} \frac{DCG_k^u}{IDCG_k^u}$$

$$DCG_k^u = \sum_{i=1}^{k} \frac{2^{R_{ui}} - 1}{log_2(1+i)}$$

Where, R_{ui} is the real rating of a user u on i ranked item, U_T is the set of users in the test data, and $IDCG_k^u$ is the best possible DCG for the user u

Experiments

Dataset

- We evaluate our approach with two standard movie data sets : MovieLens and EachMovie.
- We split user ratings into four partitions. We randomly select half of users and the rest as existing users. Similarly, we randomly split items as new and existing items. Then we use partition I for training and partition II, III and IV for test.

Experiments

Dataset

Table 1: Basic statistics of the MovieLens and EachMovie data.

	MovieLens	EachMovie
Range of ratings	1 ~ 5	1 ~ 6
# of users	6040	61131
# of items	3706	1622
# of ratings	1000209	2559107
Average Rating	3.58	4.34

Experiments

Dataset

User attributes and movie features in MovieLens and EachMovie

		MovieLens	EachMovie
User Attributes	# of gender	2	3
	# of age	7	8
	# of occupation	21	0
	# of location	0	3
	constant feature	1	1
Movie Features	# of year	13	0
	# of genre	18	10
	# of status	0	2
	# from filterbots	12	14
	constant term	1	1

Testing procedure

- In addition to the item features, 14 different filterbots are used as item features. These bots rate all or most of the items algorithmically according to attributes or users.
- For ex: An actor bot would rate all or most of the items in the database according to whether a specific actor was present in the movie or not.
- For each user in the partitions , we clustered items based on ratings given by each user.
- We considered only the items rated by the user and randomly sampled one item for each cluster. In such a way each test user is associated with a set of sampled items with size from two-five and with different ratings
- Then the nDCG is calculated.

Experiments

Cold-start setting	Algorithm	MovieLens		EachMovie	
		$nDCG_1$	STD	$nDCG_1$	STD
Existing item recommendation for new users	Random	0.4163	0.0068	0.4553	0.0055
	MP	0.6808	0.0083	0.6798	0.0166
	SMP	0.6803	0.0078	**0.6868**	0.0146
	Affinity1	0.6800	0.0077	0.6698	0.0134
	Affinity2	0.4548	0.0091	0.5442	0.0154
	Pairwise	**0.6888**	0.0078	0.6853	0.0149
New item recommendation for existing users	Random	0.4158	0.0059	0.4539	0.0052
	Affinity2	0.4489	0.0094	0.5215	0.0149
	Pairwise	**0.4972**	0.0145	**0.5821**	0.0176
New item recommendation for new users	Random	0.4154	0.0065	0.4540	0.0046
	Affinity2	0.4439	0.0102	0.5212	0.0145
	Pairwise	**0.4955**	0.0141	**0.5821**	0.0172

Summary

1. We developed hybrid approaches that not only exploit user ratings but also features of users and items for cold-start recommendation.
2. Constructed profiles for user/item pairs by outer product of individual features, and built predictive models in regression framework on pairwise user preference.
3. The unique solution is found by solving a convex optimization and resulting algorithm scales efficiently as large datasets.

Advantages

- The design offers a unified framework of collaborative and content based filtering for learning user-item affinity from all data simultaneously.
- Algorithm's efficiency scales linearly with the number of observations and hence it the application is not limited to only large datasets
- The target weight matrix is solved as a convex optimization problem and therefore the possibility of multiple values of minima does not exist.

Drawbacks

- Problem of dependency on the quality of feature generation and selection continues to exist
- Item space is **static** i.e temporal characteristics here are not updated. This can lead to incorrect recommendations.

References

- https://(https://en.wikipedia.org/wiki/Discounted_cumulative_gain) en.wikipedia.org/wiki/Discounted_cumulative_gain#Normalized_DCG(https://en.wikipedia.org/wiki/Discounted_cumulative_gain)
- http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python(http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/) /(http://www.quuxlabs.com/blog/2010/09/matrix-factorization-a-simple-tutorial-and-implementation-in-python/)
- http://(http://www.slideshare.net/DKALab/collaborativefilteringfactorization) www.slideshare.net/DKALab/collaborativefilteringfactorization(http://www.slideshare.net/DKALab/collaborativefiltering factorization)
- Basu , Chumki , Haym Hirsh, and William Cohen. "Recommendation as classification: Using social and content-based information in recommendation." Aaai / iaai . 1998.
- Chu , Wei, and Seung-Taek Park. "Personalized recommendation on dynamic content using predictive bilinear models." Proceedings of the 18th international conference on World wide web . ACM, 2009 .

CROSS-DOMAIN COLLABORATIVE FILTERING OVER TIME

LUCKY OKEHIGBEMEN

CROSS-DOMAIN COLLABORATION FILTERING OVEER TIME

OUTLINES/CONTENT

1. HYPOTHESES
2. STATE OF ART
3. PROPOSED METHOD
4. MOTIVATIONS
5. MEANING OF DOMAIN(S)
6. FRAMEWORK FORMULATION
7. ALGORITHMS
8. EXPERIMENT
9. EVALUATION
10. CONCLUSION
11. REFERENCE

HYPOTHESIS

- Does RMGM-OT model/approach provides better testing of performance level than other related others related approaches/models?
- To what extend does RMGM-OT achieve performance level than other related approaches/model?
- To what extends does RMGM-OT covers user- interest drift over time?
- Does RMGM-OT contributes in solving cold start problem?

STATE OF THE ART

This paper draw strength from several related scholars' works. And thus this has helped in great measures in putting this paper in the state of art. Some of the work the authors' drew strength from includes the following:

- Latent factor Model and Probabilistic Latent Semantic Analysis applied towards collaborative filtering (Hofmann, 1999)
- Temporal and evolutionary collaborative Filtering methods and the most well-known work in this line is the TimesSVD++ (Koraen, 2009)
- Transfer of Learning concept (Li et al 2009)
- User- interface tracking in collaborative filtering (Ma et al 2007)

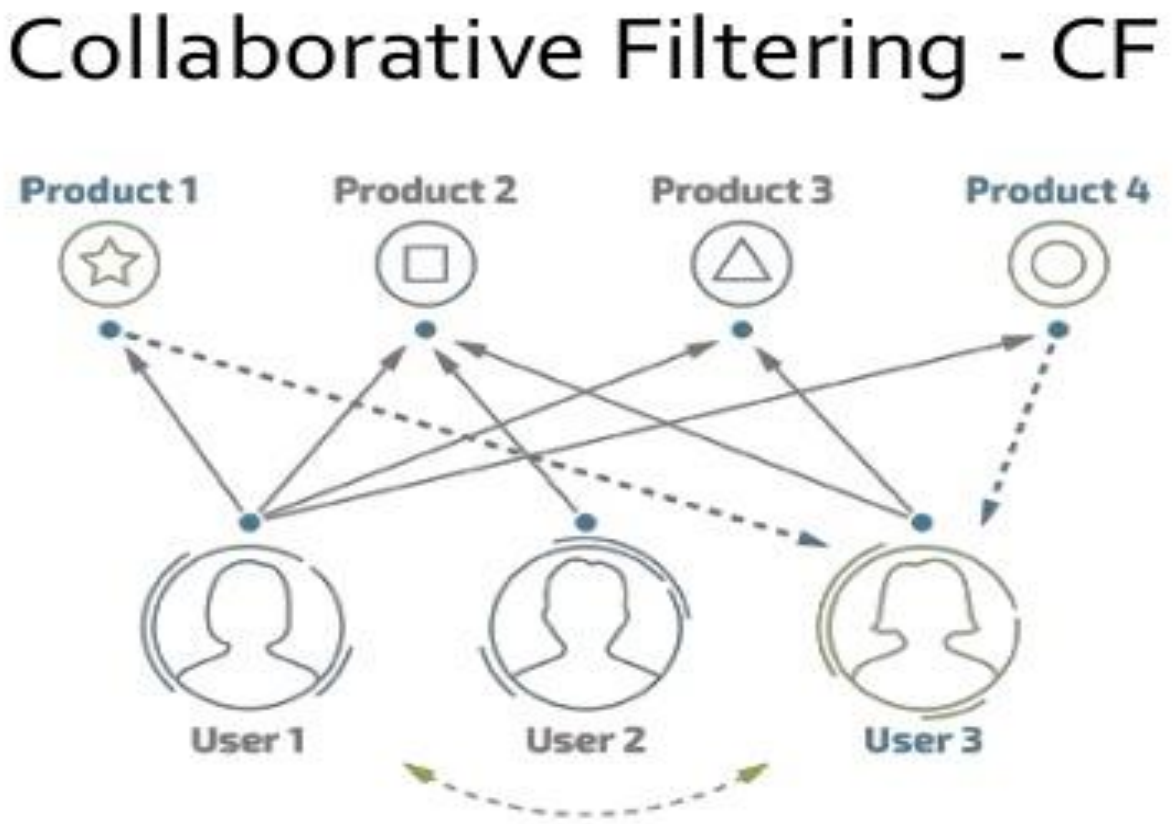
PROPOSED METHOD

- The proposed method for this paper is rating Matrix Generative Method over time. RMGM-OT

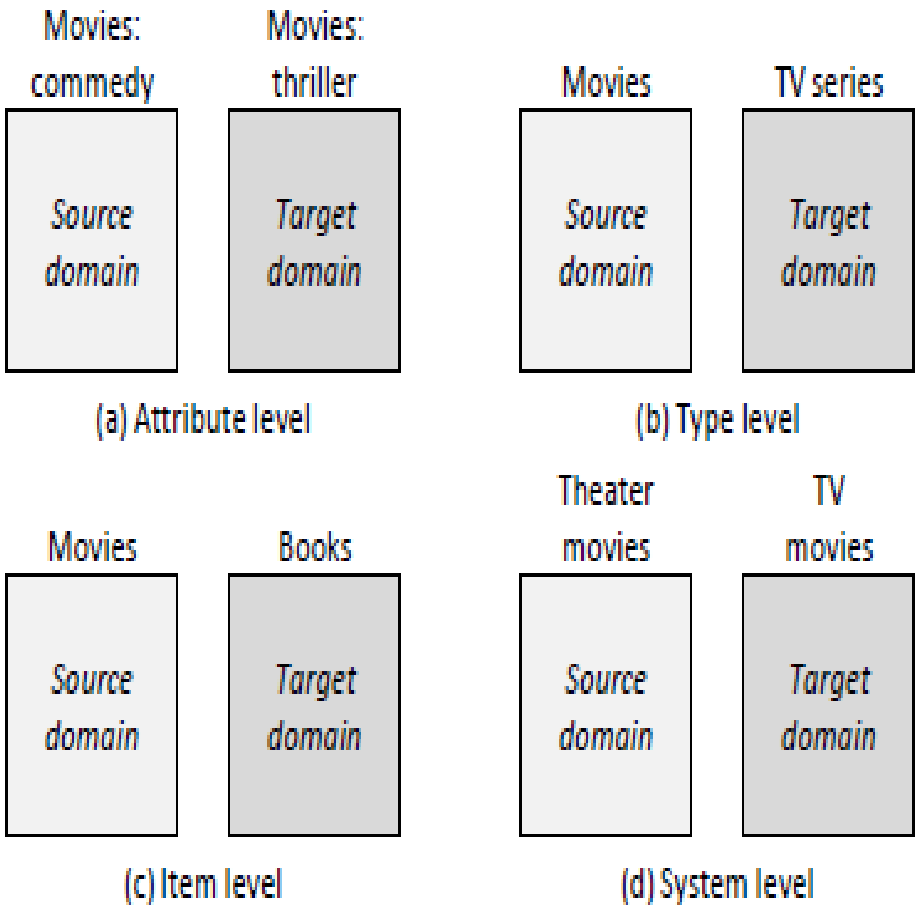
MOTIVATION OF THIS PAPER

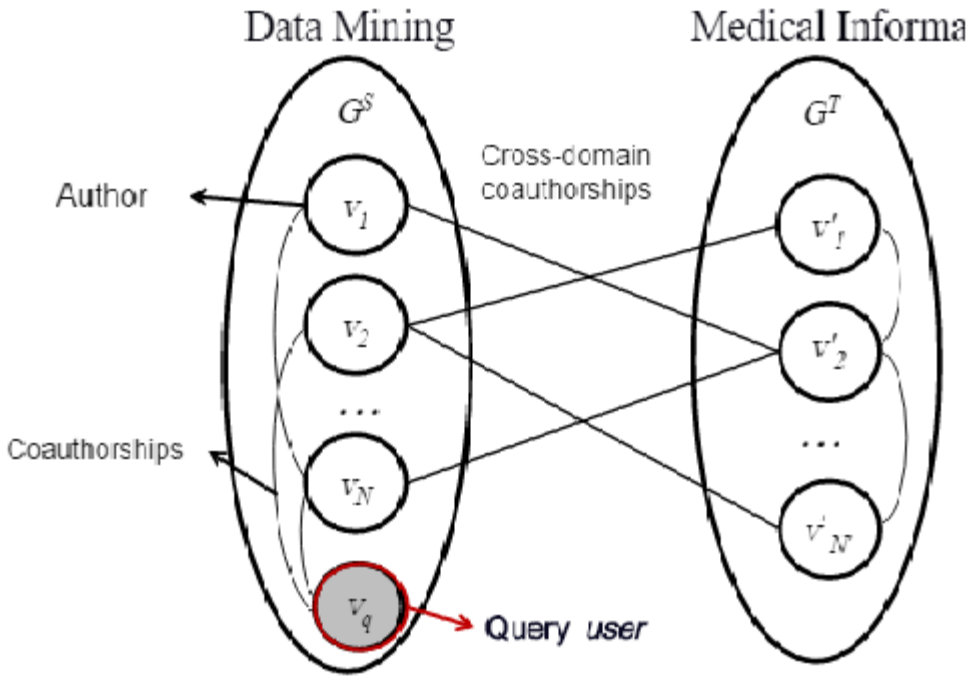
The motivation of this paper is to model user-interest drift over time based on the assumption that each user has multiple counterparts across temporal domains and the counterparts of successive temporal domains are closely related.

MEANING OF CONCEPTS

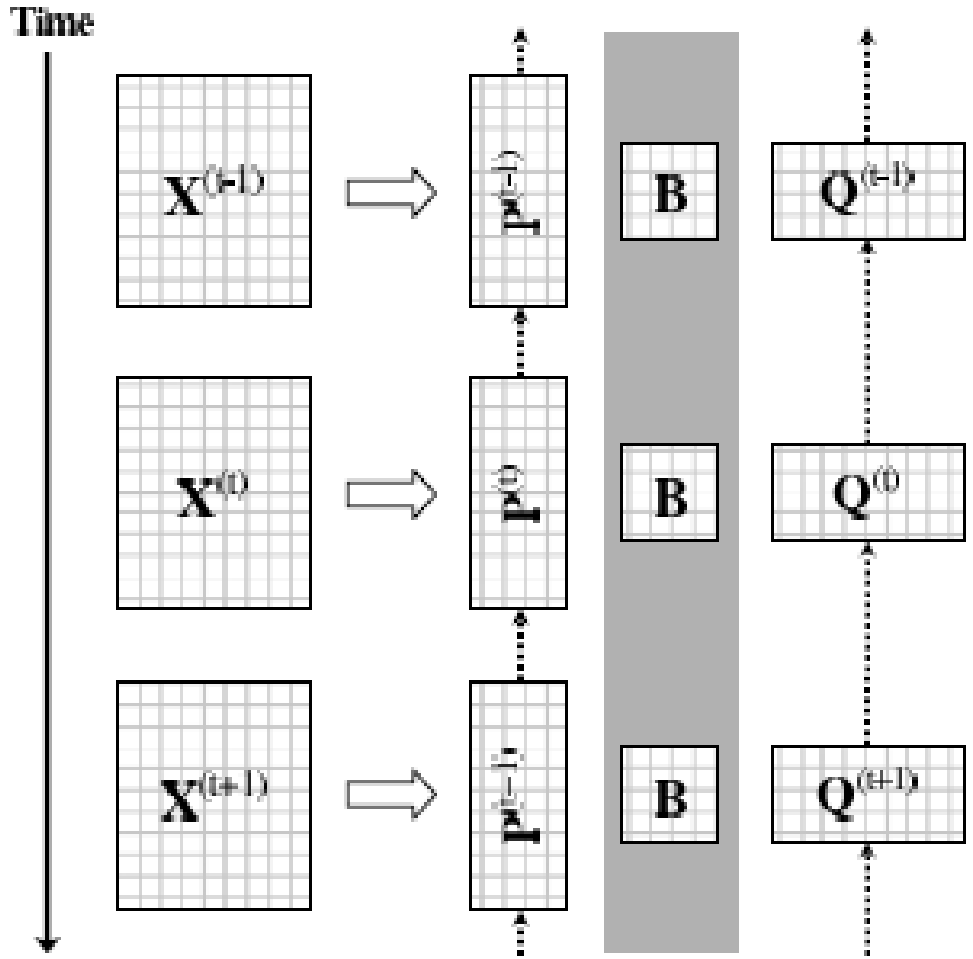


CROSS-DOMAIN





Framework for Temporal-Domain CF



Framework Formulation

- Each items in Bkl $denotes\ the\ expecting\ rating\ provided\ by\ users \rightarrow prototype\ K$ on the items $prototype\ L$
- $k\ X\ l$ prototype B learned from X
- $U_i^{(t)} \rightarrow P_i^{(t)}$ and $\sum k\ P_i^{(t)}$ [k] = 1. similarly,
- $V_j^{(t)} \rightarrow q_i^{(t)}$ *and* $\sum l\ q_j^{(t)}$ [l] = 1.
- rating $X^{(t)}$ can be predicted $yhat_{ij}^{(t)} = [P_i^{(t)}]T\ Bq_j^{(t)}$
- $q_i^{(t)}$ & $P_i^{(t)}$

Algorithms for RMGM OT

u

Algorithms for RMGM OT

Input : An N x M rating Matrix X: the Number of users and items cluster k and l

Output: A k x l prototype B learned from X

1: Randomly initialize $X^{(t+1)}$, $P^{(t+1)}$, $Q^{(t+1)}$

2: for t ← 1,……T do

3: Update $X^{(t-1)}$, $P^{(t-1)}$, $Q^{(t-1)}$

obtain $X^{(t)}$, $P^{(t)}$, $Q^{(t)}$

4: end for

5: allocates spaces for P and Q

6: for i ← 1, … … . . n do

7: j = arg max j ∈ (1…….k) $\{P_i^{(t)}\}$

8: $\sum k\, P_i^{(t)}$ [k] ← 1; $\sum k\, P_i^{(t)}$ [k] ← -1: for j ∈ (1…….k)

9: end for

10: for i ← 1, … … . . m do

11: j = arg max j ∈ (1…….l){, $q_i^{(t)}$}

12: $\sum l\, q_j^{(t)}$ [l] ← 1; $\sum l\, q_j^{(t)}$ [l] ← -1; for j ∈ (1…….l)

13: end for

14: calculate for Prototype B

Experiment

The experiment covers Netflix prize data and the details are given below

- Data preparation
- Total of 100 million rating provided
- users, ~480,000
- movies~17,000
- Time span: 1999-2005
- Each rating is associated with time-stamp

Experiment Methodology

- To cover the drifting of users interest over time, the following methodology was applied:
  - N time slices = {Jan 2002-----dec2005}
  - Time slice = 16, and each Time slice corresponded to extract 3 months
  - Time slice index {1……16}
- To obtain total **users** for the experiment
- Went further to select users with more 100 ratings in total and have at least 15 ratings in 4 Time slice. (based on the time –stamp of their first/last rating)
- Output obtain :
  - 6784 users
  - This was the total number of **users** used in the experiments

Experiment Methodology

To obtain total **movies** for the experiment

- Consideration was on movies which were imported into Netflix before 2002.
- Went further to select movies with associated rating are more 50
- Result obtain
- A total of 3287 movies

This was the number of movies used in the experiment

In evaluation

- we obtain 6784*3287 rating matrix (whose are elements are associated to 16 time slices (temporal domain))
- 4 temporal domains were used for the validation while the other 12 temporal domain were used for evaluation.

Evaluation Protocol

- The performance matrix was obtain with Root Mean Squared Error (RMSE): $\sqrt{\sum_{i \in S}(r_i - \hat{r}_i)^2 / |S|}$,

- In the evaluation, a randomly extracted 20% from the evaluation data was done and it was set for training (density 0.9%). While the remaining ratings were used for testing. This procedure iterate 5 times. The result is depicted in the table below:

Methods	RMSE (Mean±Std)
WLRA	0.942±0.001
Bi-LDA	0.918±0.004
TimeSVD++	0.929±0.002
RMGM-OT	0.919±0.002

Result Interpretation

- Addressing the system cold-start problem (system bootstrapping).
- Addressing the new user problem.
- Addressing the new item problem (cross-selling of products).
- Improving accuracy (by reducing Sparsity).
- Improving diversity.

Conclusion

- The evaluation protocol and experiment shows that this paper indeed capture transfer of learning between successive related temporal domain and achieve state of the art in recommendation performance.

References:

Thomas Hofmann (1999: pages *289-296*), **Probabilistic Latent Semantic analysis**: *In Proc of the 15th conference on Uncertainty in Artificial Intelligence.*

N. Lathia, S. Hailes, and L. Capra (2009), **Temporal Collaborative Filtering With Adaptive Neighbourhoods** (Extended Version): *In Research Note RN/09/03, Dept. of Computer Science, University College London, London, UK, April.*

Gui-Rong Xue, Chenxi Lin, Qiang Yang, Wensi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen (2005: pages 114–121), **Scalable collaborative filtering using cluster-based smoothing**. *In Proc. of the 28th SIGIR Conf.*

References:

Bin Li, Qiang Yang and Xiang yang Xue (2009: Pages 2052–2057) **Can movies and books collaborate? Cross-Domain Collaborative Filtering for sparsity reduction**. In: *Proceedings of Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI-09)*.

Ivan Cantador, Ignacio Fernandez-Tobiaz, Schlomo berkovsky, Paolo Cremonies (2012: pages187-198), **Cross-Domain Recommender Systems**: *A Survey of the start of the Art: 2nd Spanish Conferenece on information retrieval*.

Similarities

- All three are hybrid approaches for CF and CBF
- Movie-NSVD1 and Pairwise Preference Regression both make use of metadata

Differences

Movie-NSDV1

- Adresses just one cold start scenario: Recommend new movies to existing users
- Predict ratings based on user and item metadata (Generalized approach)
- Uses RMSE as performance metric
- Make use of matrix factorization approach
- Scales with metadata

Pairwise Preference Regression

- Adresses all cold start scenarios
- Uses user ratings and user and item metadata
- Uses nDCG as performance metric
- Make use of outerproduct of tensors
- Scales with number of observations (ratings)

RMGM

- Adresses all cold start scenarios
- Assumes that each user has multiple counterparts over temporal domains and sucessive counterparts are closely related
- Uses RMSE as performance metric
- Make use of Bayesian latent factor model for matrix tri-factorization

Winning method

- Cross domain filtering is our winning method. It solves the cold start problem by inferring users preferences from a source domain to a target domain.

Backup: Presentation 1

Experiment – MF vs. Movie-NSVD1

Use Netflix Prize Dataset „Probe“

Train on a set of 140 840 ratings –“Probe10" (1/10 of Probe)

Predict values on the evaluation set – „Quiz“ with 2817 131 ratings

Collect movie metatdata and represent texts as vectors

Use following zones: title, synopsis, actor, director, release year, genre.

Movies are described with 146 810 dimensional vectors

Dimensional vectors e.g.: genre:comedy, title:comedy

81 nonzero values on average

Learning algorithms for MF

IGD01 - Incremental gradient descent with bias bu and ci

- RMSE: 0,9079

AG01: like IGD01, but with alternating gradient descent

- RMSE: 0,9096

ALS01: MF with biases bu and ci, using alternating least squares, starting with movie features

- RMSE: 0,9257

Hyperparameter

- Define higher level concepts about the model such as complexity, or capacity to learn.
- Cannot be learned directly from the data in the standard model training process and need to be predefined.
- Can be decided by setting different values, training different models, and choosing the values that test better

Examples:

- Number of latent factors in a matrix factorization
- Learning rate (in many models)

/streichenMovie-NSVD1: Batch training algorithm

```
1  Create random model.
2  repeat
3      one epoch:
4      Q ← XW
5      foreach (u, i) ∈ R do
6          update b_u, c_i, p_u and q_i as in
7          Algorithm 1, lines 8–14.
8      end
9      // now update W such that the actual value of
10     // Q is better approximated by XW
11     for 1 to n_2 do
12         for i ← 1 to M do
13             e ← q_i − W^T x_i
14             d ← 1/(x_i^T x_i)
15             W ← W + η_2 · (d · x_i · e^T − λ_2 · W)
16         end
17     end
18     Q ← XW
19 until RMSE on the test set decreases ;
```

Algorithm 2: Batch training algorithm for movie-NSVD1

New parameters:

- η_2: learning rate
- λ_2: regularization factor
- n_2: iteration count

Time requirement: $O(K * n2 * |X| + K * |R|)$

Predicting ratings on new movies

- CBF has advantage over CF: ability to handle new movies
- New movies have no ratings
- But metadata is available
- **Possible Scenarios**
  - If descriptions of movies are as valuable as their ratings, then X10 RMSE should be equal to Probe10 RMSE where movies has many ratings
  - If only metadata is unhelpful in predicting user preference, then X10 RMSE should be not much better than that of a user-bias-based method (prediction is based on the average rating of active user)

Recommender Systems

- Use opinions and preferences of users to help others find useful and interesting content
- Personalize what users see
- Web applications which involve predicting user responses to options

Philosophy:

- Makes use of the social process of friends endorsing items as recommendations to others in their community. Recommender systems automate this process.

Applications:

- Offering news articles to on-line newspaper readers, based on a prediction of reader interests.
- Online e-commerce applications like Amazon to improve user engagement and loyalty

Similarities

- All three are hybrid approaches for CF and CBF
- None of these approaches make use of feature generation and selection
- Movie-NSVD1 and Pairwise Preference Regression both make use of metadata